UNIVERSIDADE DE LISBOA

FACULDADE DE CIÊNCIAS

DEPARTAMENTO DE INFORMÁTICA

Assistente Virtual para Fatura Inteligente

José Pedro Franco Rodrigues

Mestrado em Engenharia Informática
Especialização em Engenharia de Software

Versão Pública

Trabalho de Projeto Orientado por:
Professor Doutor Carlos Lourenço

2019

Agradecimentos

Em primeiro lugar, quero agradecer aos meus Pais, por todo o apoio que sempre me deram ao longo da minha vida académica e meu crescimento pessoal. Foram sempre um dos meus pilares e a uma das minhas principais fontes de motivação para eu atingir os meus objetivos. Obrigado Mãe e Pai.

Ao meu irmão David, à minha família, um enorme obrigado por terem estado sempre presentes a dar-me força. Um agradecimento muito especial aos meus queridos Avós, sempre foram e serão importantes na minha vida! Obrigado pela vossa ajuda e carinho!

Quero agradecer aos meus amigos, aos que já me acompanham desde a minha adolescência, muito em especial ao Carlos Mota e Marcelo Carvalho, e aos que fui abraçando e partilhando momentos ao longo da minha vida académica. Se há pessoas com quem sempre pude contar nos bons e maus momentos foram vocês! Em especial, gostaria de agradecer à minha namorada Maria Constança.

Quero agradecer aos meus colegas de faculdade, que partilharam comigo todo este caminho percorrido até aqui. Em especial, ao Alexandre Rocha, André Krippahl, Paulo Querido e Tiago Marques, por todos os momentos passados com vocês dentro e fora da Faculdade!

Quero agradecer à FCUL o acolhimento e apoio durante todo o meu percurso académico.

Quero agradecer à minha colega de estágio Liliana Ramos, foi um prazer ter trabalhado contigo durante estes 9 meses.

Quero agradecer ao Bruno Santos e Marta Almeida, por toda ajuda e paciência que tiveram comigo durante praticamente todo o meu percurso neste estágio na Accenture Portugal. Foi espetacular ter partilhado esta experiência com vocês!

Quero agradecer ao meu orientador na empresa, Nuno Ferreira, por me ter permitido experienciar este estágio e pela ajuda e confiança que sempre demonstrou e depositou em mim ao longo de todo o meu percurso até hoje na Accenture Portugal.

Quero agradecer ao meu orientador da Faculdade, Professor Carlos Lourenço pela ajuda prestada no desenvolvimento deste documento.

Por fim, mas não menos importante, gostaria de agradecer a todas as pessoas da Accenture Portugal, que estiveram comigo neste estágio, em especial, à Cátia, Beirão, Artur, Ruben, Jorge, Joana, Tomás, Paulo e Pedro Eugénio, que permitiram ter um ambiente espetacular dentro da Accenture Portugal.

Aos meus Pais e aos meus Avós.

Resumo

No âmbito do projeto realizado na empresa *Accenture Portugal*, durante o estágio de 9 meses, iniciado a 15 de Outubro de 2018, para a conclusão do Mestrado de Engenharia Informática com especialização em Engenharia de Software, apresento o meu relatório.

Este projeto visa a criação de uma aplicação, denominada de "*Brainy Telco Bill",* que suporta um Assistente Virtual, com o propósito de criar mais uma alternativa de apoio ao Cliente, de forma a aliviar os *call-centers* e espaços de apoio ao mesmo em empresas ligadas ao sector das telecomunicações. A aplicação "*Brainy Telco Bill"* é composta por dois módulos, um, o Assistente Virtual e o outro, o *front-end* que possibilitam ao Cliente em conjunto, a visualização das suas faturas de forma mais interativa e intuitiva, enquanto comunica com o Assistente Virtual.

Este Assistente Virtual é responsável por receber as dúvidas dos Clientes e pedidos de informação, bem como efetuar a resposta aos mesmos.

Para interpretar as mensagens do Cliente, o Assistente Virtual, utiliza algoritmos de análise de dados, com o objetivo de compreender qual a intenção do Cliente. Após a mensagem ser processada, responde à dúvida ou ao pedido com base na *intent* e nas *entities* (informações e conteúdos importantes na frase, com relevância para o Assistente Virtual) identificadas na frase.

Para que o Assistente Virtual consiga compreender a linguagem natural é utilizado um processador de linguagem natural, sendo o mesmo disponibilizado pela Microsoft, para que continuamente o Assistente auxilie o Cliente ao nível de questões de faturação. Todas as mensagens recebidas são usadas para enriquecer a base de conhecimento do Assistente Virtual.

Assim, para que este projeto seja bem gerido, iremos utilizar uma metodologia de desenvolvimento de software, chamado *V-Model*. Este Modelo é uma extensão do modelo *Waterfall*, onde a cada fase do ciclo de desenvolvimento corresponde uma fase de testes. [20,21]

Palavras-chave: Assistente/Agente/Assistente Virtual, Linguagem Natural (NLP), *Machine Learning, Fatura.*

Abstract

In scope of the project developed at Accenture Portugal, during a 9-month internship which started at October 15, 2018, I am writing this essay to finish the Master in Software Engineering.

This project aims to create an application, named “Brainy Telco Bill”, for a telecommunications company, who supports a chatbot with the purpose of creating an alternative to customer support, in order to lessen the work of call-center and customer support spaces. The application is composed in two models, one for a Chatbot and another one for a front-end that gives the client the possibility to visualize his invoice and talk with the virtual agent at the same time.

This Agent (Virtual Assistant) receives messages from the client and will try to find the intentions and entities of the messages received. Once the intent of the client's message is detected, the agent can correctly answer the customer's question or assertion.

To be able to understand natural language and identify the intents and entities, the Chatbot, uses a natural language processor, provided by Microsoft. Using machine learning and a data analysis can also enrich his knowledge of data. In addition, through external APIs it is possible to access customer's billing information and solve the customer's question or problem successfully.

We will use a methodology for software development processes, called V-Model, so we can organize our data. This Model is an extension of the Waterfall model, where every phase in the software development cycle has a corresponding phase in the testing cycle. [20,21]

Keywords: Chatbot, Bot, Agent, Natural Language (NLP), Machine Learning

Conteúdo

Lista de Figuras

Lista de Acrónimos

NLP – *Natural Language Processing*

LUIS – *Language Understanding*

Paas – *Platfrom as a Service*

Saas – *Software as a Service*

AOP – *Aspect Oriented Programming*

API – *Aplication Programming Interface*

REST – *Representational State Transfer*

HTTP – *Hypertext Transfer Protocol*

NPM – *Node.js Package Manager*

HCI – *Human Computer Interaction*

JSON – *JavaScript Object Notation*

Capítulo 1

Introdução

Com a constante evolução das capacidades comunicativas do Homem, foram criadas novas formas de comunicação, seja por carta, rádio, telemóvel e mais recentemente pela internet. A comunicação é uma das necessidades do ser humano, que por natureza é um ser social.

A interação humano-computador (HCI) nos últimos 30 anos apresentou também uma grande evolução a nível da *User Experience*. As pessoas utilizam uma linguagem natural para a comunicar entre si, da mesma forma o humano usa uma linguagem natural para comunicar com o computador/ Assistente Virtual. [4]

Este foi um dos grandes *inputs* por detrás do desenvolvimento dos assistentes virtuais. Um Assistente Virtual é um software que permite a interação entre um ser humano e um computador.

Quando os primeiros exemplos de assistentes virtuais foram criados, não tinham ainda finalidades definidas, apenas eram desenvolvidos por diversão e as técnicas usadas eram relativamente simples. Por exemplo, em 1966 a *ELIZA* utilizava uma técnica baseada na correspondência de palavras-chave do texto introduzido pelo o utilizador. Mais tarde, em 1995, foi criada a ALICE, que com técnicas de *pattern matching* foi um dos principais assistentes a ser utilizado. [3]

Com a rápida evolução da tecnologia, surgiu a Siri (Apple) e o Google Now (Google) como todos conhecemos hoje. Na figura 1 é ilustrado a evolução dos assistentes virtuais desde o primeiro a ser criado até aos que existem na atualidade.

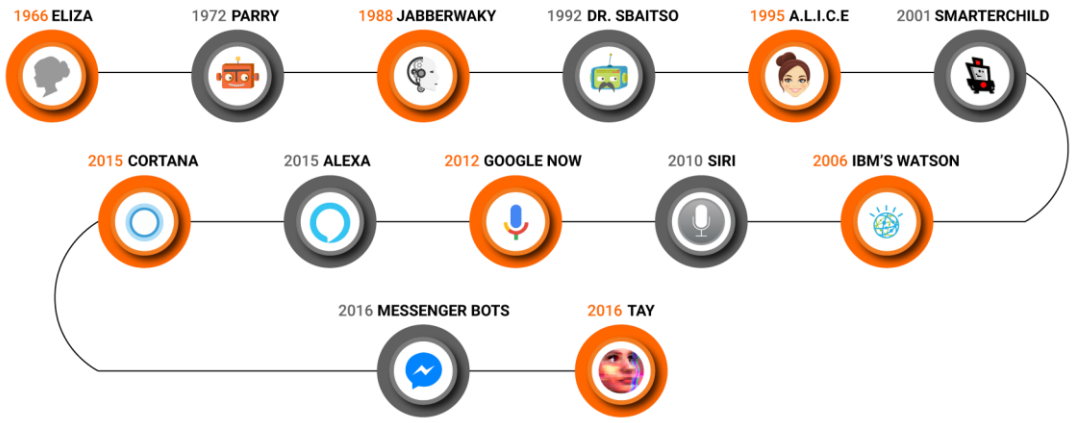

Figura 1 - História dos Assistentes Virtuais

Mais recentemente, foram criadas diversas plataformas de desenvolvimento para auxiliar os programadores a desenvolver os seus assistentes virtuais. Neste sentido a *Accenture Portugal*, percebeu que a necessidade de criação de um Assistente para a área de faturação seria uma mais valia.

Assim, foi projetado o desenvolvimento de uma aplicação web, onde será integrado um Assistente Virtual. Este assistente não só terá mecanismos de interpretação de linguagem natural, como também ferramentas de *machine learning*, com o objetivo de ao longo do tempo, alargar a base de conhecimento do Assistente Virtual e assim conseguir responder cada vez mais, a mais questões que lhe podem ser colocadas.

Este projeto tem como finalidade a criação de um Assistente Virtual, para auxiliar as Empresas de Telecomunicações no âmbito do apoio ao Cliente, para tal é fundamental que se consiga fazer a simulação de uma conversação natural entre o ser humano e o computador, de forma a criar uma maior qualidade de serviço e de experiência para ao utilizador. Assim, será possível reduzir o número elevado de filas de espera e de chamadas nos *call centers*. [1]

1.1 Motivação

As grandes empresas, quer privadas, quer públicas, prestadoras de serviços, deparam-se cada vez mais com um elevado incremento de pedidos por parte do Cliente, seja em *call centers* ou em locais de apoio ao Cliente. Por mais que tentem otimizar os processos de atendimento ao Cliente, através de novos modelos de decisão, o tempo de espera e a sobrecarga das filas de espera continuam a aumentar. [1]

Muitas destas empresas, são Clientes da *Accenture Portugal*, daí a necessidade de desenvolvimento deste projeto.

Com base na evolução das tecnologias na área inovação, cada vez mais é exigido, face à concorrência, que sejam desenvolvidos novos sistemas inovadores, como é o caso de Assistentes Virtuais.

A criação de um Assistente Virtual pode ser uma das melhores soluções para agilizar este problema. Os Clientes através desta solução poderão obter as respostas às suas dúvidas sem a necessidade de nenhum outro contacto direto ao apoio do Cliente e resolvendo desta forma, as dúvidas que lhe foram suscitadas de uma forma mais célere.

1.2 Objetivos

Os objetivos estabelecidos para este estágio foram:

1. **A criação de um Assistente Virtual que interprete as mensagens recebidas do Cliente, através do recurso a ferramentas de interpretação de linguagem natural** – Com a realização desta tarefa foi possível criar a base principal do Assistente Virtual, pois um interpretador de linguagem natural é um dos pilares mais importantes e fundamentais. Esta tarefa numa primeira fase teve como objetivo a modelação do NLP e posteriormente a incorporação com a lógica aplicacional.
2. **O desenvolvimento de toda a lógica de negócio ligada à análise das faturas do Cliente** – A lógica de negócio para além de tratar do fluxo, do contexto e da sessão dos utilizadores, também tem como principal objetivo o desenvolvimento de algoritmos de análise de faturação, para que o Assistente Virtual tenha uma maior capacidade de explicar e esclarecer as dúvidas do Cliente sobre a sua fatura.
3. **Criação de aspetos para efeito de *logging*** – Esta tarefa é necessária para guardar e registar na base de dados todas as mensagens trocadas entre o Assistente Virtual e o Cliente, para que no futuro possam ser utilizadas para múltiplos objetivos.
4. **Integração do Assistente Virtual com o *front-end*** – Depois de ser realizado o *deployment* do Assistente Virtual para o serviço *cloud Azure*, é necessário criar canais de comunicação para que o Assistente Virtual seja integrado com o *front-end*.

1.3 Contribuições

Este projeto foi desenvolvido por uma equipa de três pessoas, com uma metodologia V-Model. A minha contribuição foi baseada nas etapas de desenvolvimento, referentes

ao Assistente Virtual, nomeadamente a análise funcional, o levantamento de requisitos, a implementação e por último a fase de testes ao sistema.

De acordo com a metodologia usada, integrei a equipa nas fases de levantamento dos requisitos, de desenho do sistema e da arquitetura. Onde, de uma forma transversal, estive em contacto com pontos comuns entre a aplicação de *frontend* e o Assistente Virtual.

Na fase de implementação, participei nas seguintes vertentes da aplicação:

- Desenvolvimento da lógica de avaliação e cálculos sobre as faturas do Cliente.
- Desenvolvimento do *chatbot*, ou seja, a aplicação que contêm toda a lógica de negócio de fluxos e decisões acerca da conversação entre o Assistente Virtual e o Cliente.
- Modelização e integração do NLP para a interpretação das mensagens do Cliente.
- Decisões sobre *tokens* enviados pelo *frontend* para o *bot* que indicam ações concretas.
- Desenvolvimento da camada de persistência de dados.
- E por fim a integração do Assistente Virtual no frontend.

Durante a execução do projeto e mesmo na fase de testes, foram criados dois canais que faziam a ligação do Assistente Virtual ao Skype e ao *frontend*, permitindo receber feedback dos vários colaboradores da empresa. Para além destes testes unitários e integrados (*chatbot* e *frontend*), foram realizados testes ao NLP através da plataforma da *framework* da Microsoft, LUIS, que contêm uma área de testes.

1.4 Estrutura do documento

Este documento está organizado da seguinte forma:

- Capítulo 1 – Neste capítulo, é feita uma introdução ao tema, bem como a motivação que existiu para a realização do mesmo, são explicados quais foram os objetivos, as contribuições e a estrutura deste documento.
- Capítulo 2 – Este capítulo intitula-se de trabalho relacionado, onde é pretendido apresentar o estado de arte e a explicação das tecnologias e *frameworks* usadas. Através da comparação entre diferentes e possíveis soluções para o mesmo problema, é justificado um conjunto de decisões tomadas a nível das tecnologias e *frameworks* usadas.

- Capítulo 3 – Neste capítulo, é apresentado o plano de trabalho realizado com base na metodologia usada e são também apresentados os requisitos e casos de uso definidos para o projeto.
- Capítulo 4 – Capítulo, intitulado de Desenho, onde são apresentadas as diferentes perspetivas de arquitetura da plataforma, as tecnologias utilizadas no desenvolvimento do Assistente Virtual e as decisões tomadas para a arquitetura implementada e para a as tecnologias utilizadas no desenvolvimento do projeto.
- Capítulo 5 – Capítulo, intitulado de Desenvolvimento, onde é explicado todo o trabalho realizado no desenvolvimento do Assistente Virtual.
- Capítulo 6 – Capítulo, onde são apresentados todos os testes realizados ao sistema, com base na metodologia usada para o desenvolvimento.
- Capítulo 7 – A conclusão, composta por 3 subcapítulos: dificuldades encontradas, conclusão do trabalho realizado e por fim o trabalho futuro, é também acompanhado pelas decisões tomadas previamente para a adaptação de possíveis alterações futuras.
- Bibliografia.

Capítulo 2 Trabalho relacionado

2.1 Estado de Arte

Este Projeto realizado na *Accenture Portugal,* visa a criação de um Assistente Virtual, com capacidades de *machine learning* e *data analytics,* com o objetivo de ser integrado numa aplicação web onde promoverá o contacto direto com o Cliente.

As diferentes abordagens, foram tomadas com o objetivo de criar/implementar este Assistente Virtual, exemplo para a utilização de uma determinada tecnologia ao invés de outras que têm diferentes custos e benefícios.

Uma das abordagens possíveis, poderia ter sido o desenvolvimento de um Assistente Virtual totalmente à medida. Esta abordagem pressupunha o desenvolvimento de um Processador de linguagem Natural, como também todos os módulos necessários para a sua implementação. Mas esta abordagem tornou-se inviável na medida em que existem outras opções mais rentáveis.

Deste modo, para o desenvolvimento deste *chatbot* foram analisadas quais seriam as melhores *frameworks* para desenvolver os diferentes módulos do projeto.

O mercado oferece muitas alternativas, como por exemplo: *Microsoft Bot Framework*, *DialogFlow*, *IBM Watson* ou *Amazon Lex*, mas todas elas têm prós e contras.

Após várias análises as estas alternativas, para a escolha da melhor plataforma de desenvolvimento do *chatbot*, percebeu-se que a escolha não deveria ser baseada na performance, nem no custo, mas sim na alternativa que melhor interpretasse linguagem natural e onde fosse possível integrar todos os serviços e componentes necessários para a elaboração da lógica de negócio relacionada com a área de faturação.

Ou seja, a melhor plataforma/ *framework* que permitisse utilização sistemas externos e que ao mesmo tempo disponibilizasse um controlo total sobre a sessão do utilizador e gestão do fluxo e contexto de conversação, seria a melhor solução.

Com os requisitos descritos acima, a melhor opção foi o *Bot Framework* da *Microsoft*, porque embora nas outras opções fosse possível desenvolver o Assistente Virtual fora da plataforma, como é o caso do *DialogFlow*, estas são consideradas como *no-code* e *low-code*, ou seja, serão sempre dependentes da plataforma onde ficam definidos os diálogos, as intenções e as entidades da aplicação. O que dificulta a integração com sistemas externos que são necessários para concretizar o nosso objetivo para este Assistente Virtual.

Para o desenvolvimento do Assistente Virtual, foi necessário também definir qual a linguagem a ser utilizada. Com base na decisão entre duas linguagens, que o *Microsoft Azure* suporta, a escolha é feita entre *Node,js* ou C#.

O C# é uma linguagem orientada a objetos desenvolvida pela Microsoft e o Node.js é uma *framework* de javacript, desenvolvida para ser executada no lado do servidor.

Foi decidido desenvolver o Assistente Virtual em Node.js, uma vez que o *javascript* tem um repositório de documentação e bibliotecas superior em relação às bibliotecas para C#.[19,20]

Dentro desta *framework*, Node.js, é possível desenvolver o Assistente Virtual em *Javascript* ou em *Typescript*. Optou-se por se desenvolver em *Typescript,* visto que é uma linguagem tipificada, ao contrário de *Javascript*, que não contêm tipos. Sendo o desenvolvimento feito por uma equipa com diferentes funções, torna-se mais simples o desenvolvimento e a compreensão do código quer por parte da equipa de programadores, quer da equipa que irá testar desenvolvimento.

A Microsoft disponibiliza diversos serviços e *frameworks* para auxílio no desenvolvimento do Assistente Virtual. O *LUIS NLP* é um interpretador de linguagem natural que o Assistente Virtual utiliza para interpretar a mensagem enviada pelo Cliente. O *Bot Framework Service* é um serviço Paas (*Platform as a Service*) *cloud* do *Microsoft Azure* onde é realizado o *deployment* do *chatbot* e onde este pode ser executado.

A comunicação entre o Assistente Virtual e o *frontend* é realizada através de uma API chamada *Direct Line*. Cada comunicação é feita através de um estilo arquitetural chamado *REST*.

2.2 Node.js e Typescript

O *Typescript* é um superconjunto de *javascript*, que quando compilado gera código *javascript* para ser interpretado em *browsers* ou mesmo corrido em servidores. Esta extensão de *javascript* é caracterizada por trazer mais organização nas relações entre os diversos componentes do código, ou seja, tudo pode ser tipificado e as ligações entre classes no que toca a herança e polimorfismo são mais facilmente compreendidos.

O *Node.js* é uma *framework* de *javascript* que não é executado do lado do Cliente, mas sim no lado do servidor. Esta alternativa foi desenvolvida para existir uma nova forma de criar aplicações que lidem melhor com a sincronização dos pedidos, ou seja, as chamadas deixam de ser bloqueantes e os pedidos não esperam pelo retorno de outros para continuar a sua execução. [10]

O *JavaScript* por natureza é uma linguagem síncrona e bloqueante, mas quando usamos *JavaScript* para programação no lado do servidor passa a ser assíncrona e não-bloqueante. [11,12]

Ao contrário de outras linguagens, como por exemplo Java ou Python, que tem bibliotecas bem definidas pelos seus criadores, a linguagem Javascript usa como bibliotecas um repositório partilhado, denominado de NPM. Este repositório é um projeto *open-source* criado de forma a existir uma maior partilha de código onde qualquer pessoa pode contribuir para a referida "biblioteca" de *JavaScript*.

2.3 Cloud Microsoft Azure

A *Microsoft Azure,* é uma plataforma que oferece muitos serviços *cloud*, permitindo que seja possível criar e gerir aplicações recorrendo a estes serviços. Como é um serviço em *cloud* tem bastante escalabilidade, tolerância a faltas e uma opção de poder aumentar ou diminuir o número de instâncias dedicadas à performance, que de outra forma, com servidores locais, sem um grande investimento jamais o poderíamos fazer. [5]

Como a maior parte das plataformas *cloud*, o Microsoft Azure disponibiliza diferentes tipos de modelos de cloud, nomeadamente, os modelos IaaS (Infrastructure as a Service), PaaS (Platform as a Service) e SaaS (Software as a Service). Cada um destes modelos disponibiliza um nível de abstração diferente, com o objetivo de reduzir o esforço requerido pelo utilizador no desenvolvimento de um sistema.

- IaaS – um serviço que disponibiliza a capacidade de processamento, o armazenamento, a gestão das redes e a liberdade de escolha do sistema operativo. E que abstrai toda a infraestrutura do utilizador, ou seja, o utilizador não tem que se preocupar com a infraestrutura onde irá correr o sistema operativo.

- PaaS – um serviço que oferece uma plataforma para que os utilizadores possam executar as suas aplicações na *cloud*. Ao contrário do IaaS, este serviço não precisa que os utilizadores definam o seu próprio sistema operativo.

- SaaS – uma camada de alto nível onde uma aplicação é oferecida como um serviço, onde os utilizadores não necessitam de configurar nenhuma componente da *cloud*.

2.3.1 Bot Framework e NodeJS

Esta *framework*, disponibilizada pela *Microsoft Azure*, oferece ainda vários serviços de criação de Assistentes Virtuais, seja através de um modelo Saas (*Software as a Service*) ou de um modelo Paas (*Platform as a Service*). [6]

Para complementar este produto, a Microsoft tem também dois outros serviços que ajudam na criação de um Assistente Virtual, produtos estes que ajudam na interpretação da linguagem natural e também aplicam técnicas de *machine learning* com vista a criarem conhecimento a partir dos dados que lhe são fornecidos.

Mas para entender melhor como funciona o Assistente Virtual (*Bot*) e como ele interage com o Cliente é necessário conhecermos os conceitos base. Existem diversas funcionalidades que irão facilitar o Assistente Virtual, quer em termos de comunicação com o Cliente, quer da lógica de diálogos entre o *bot* e o Cliente e quer dos interpretadores de linguagem natural que identificam as intenções e entidades da mensagem que o Cliente enviou.

O Assistente Virtual vai conter um *RESTful web service* otimizado para este objetivo de envio e receber mensagens com base numa *framework* chamada *Restify*, que foi desenvolvida para auxiliar o Node.js a realizar o seu *deployment*.

Os Conectores servem para a configuração da troca de mensagens entre o Assistente Virtual e o *Bot Framework*. [6]

Os *Dialogs* que são a estrutura de toda a lógica de conversação do Assistente Virtual com o Cliente, ou seja, é o Assistente Virtual que vai interpretar e comandar o fluxo de conversação entre ele e o Cliente. Com os *Dialogs* é permitido executar e gerir o fluxo da conversa seja ela simples ou complexa através de funções disponibilizadas por esta biblioteca, como por exemplo: beginDialog(), endDialog().[6]

Os *Recognizers* servem para o reconhecimento da intenção do Cliente quando envia uma frase, através de *frameworks* de auxílio ao Assistente Virtual chamadas LUIS e Qna Maker. Estas *frameworks* estão explicadas mais abaixo

O modo como é realizada a comunicação entre o cliente e um assistente virtual desenvolvido nesta *framework*, encontra-se descrito na figura 2. Esta figura começa por apresentar que uma mensagem é enviada através de um pedido http para um endpoint, que após receber a mensagem, encaminha para os controladores presentes no assistente virtual, com o objetivo de estes resolverem qual a resposta que deve ser retornada para o cliente. Depois desta decisão a mensagem de resposta é gerada e enviada para o cliente.

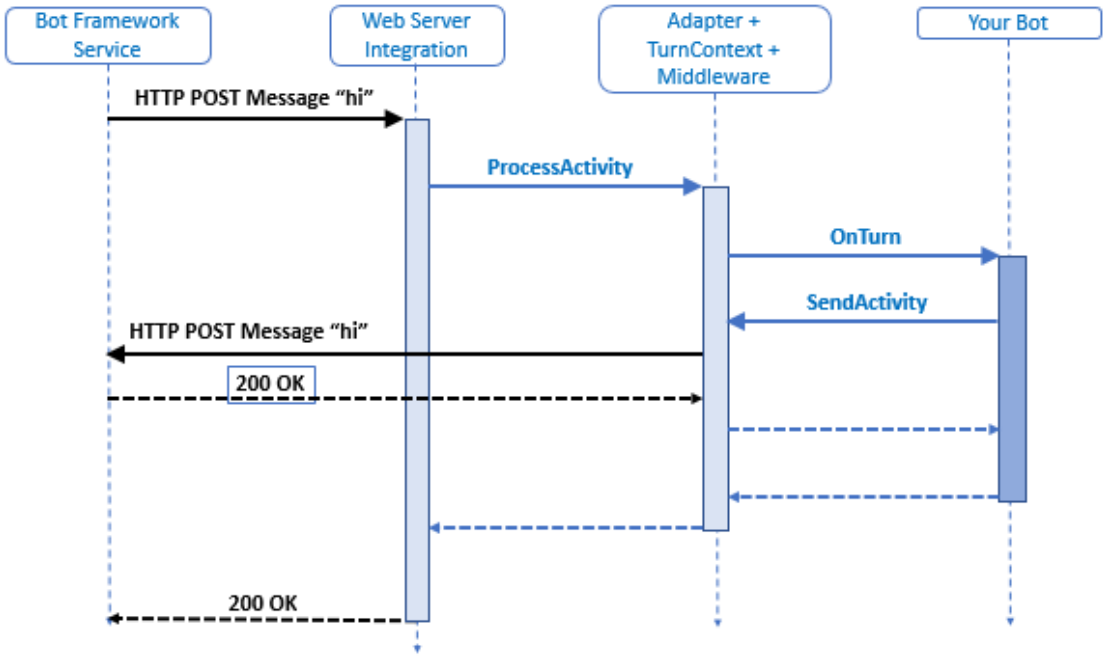

Figura 2 - Comunicação entre o Assistente Virtual e o Cliente [7]

2.3.2 Table Storage (Base de dados)

O Armazenamento do *Azure* é mais um dos serviços que está disponível na *cloud* da *Microsoft*, possui os seguintes serviços de armazenamento de dados: um Armazenamento de objetos (dados binários e texto), a partilha de ficheiros para a *cloud*, um arquivo de mensagens fiáveis entre componentes da aplicação e por fim o *Table Storage*, uma base de dados não relacional.

O *Table Storage* é uma estrutura de tabelas usada para criar dados estruturados não relacionais, ou seja, uma vez que não existem esquemas, é fácil adaptar os dados que são guardados à medida que o projeto vai evoluindo. Como se trata de uma base de dados *Nosql*, o acesso aos dados é mais eficiente e mais rápido tornando-se mais económico do que um sistema de base de dados tradicional a usar SQL.

A maneira mais fácil de entender o funcionamento de uma base dados não relacional, neste caso a *Cosmos DB*, é perceber que existem três grandes dimensões, as tabelas, as entidades e as propriedades. As tabelas guardam os dados como coleções de entidades. Uma entidade é semelhante às linhas de uma de uma tabela num modelo de base de dados relacional e é representada pela sua *primary key* e por um conjunto de propriedades, que são semelhantes às colunas. As propriedades, num contexto de uma base de dados relacional, são consideradas os atributos de uma tabela.

Uma base dados não relacional caracteriza-se por ter um escalonamento horizontal, o que evita que seja necessário existir um modelo de entidades previamente definido antes

mesmo de ser criada, isto é, sempre que é necessário acrescentar uma nova propriedade a determinadas entidades de uma tabela, essa adição é relativamente simples. Assim, é um modelo muito usado para clusters e grandes infraestruturas que precisem de escalabilidade e de grande balanceamento de carga para serem mais eficientes.

Neste sistema de base de dados, a chave primária divide-se em duas partes, respetivamente, primeira parte a *PartitionKey* e a segunda parte a *RowKey*. A primeira parte da chave primária identifica a partição em que a entidade está alocada, sendo que estas partições servem para efeitos de balancear a carga sobre as diferentes partições do sistema na *cloud*. A segunda parte da chave primária identifica a entidade dentro de cada partição. Assim, as duas chaves em conjunto conseguem, univocamente, identificar a entidade numa tabela da base dados.

Uma vez que o *chatbot* vai estar alojado na *cloud* e a quantidade de informação a ser guardada poderá ser elevada, foi decidido usar uma base de dados não relacional para responder às necessidades que o projeto necessite.

2.3.3 LUIS (Processador de Linguagem Natural)

O LUIS, é um serviço em *cloud* que utiliza *machine learning* de forma a que o Assistente Virtual a ser utilizado neste serviço seja capaz de entender linguagem natural e ao longo do tempo evolua de forma a ter cada vez mais respostas corretas às questões do Cliente. Este serviço suporta 12 línguas, podendo interpretar diferentes linguagens caso seja necessário.

O LUIS, é então o *recognizer* que pode ser integrado no *Bot Framework* de forma a identificar as intensões e entidades e assim o Assistente Virtual passa a ser capaz de responder ao Cliente de forma correta.

Em primeiro lugar é necessário modelizar este NLP, começando pela inserção das intents e entities gerais que o Assistente Virtual consiga responder. De seguida o programador terá de treinar e publicar o NLP, de forma a ser criado um http *endpoint* no *Azure*, para que este possa receber as mensagens do utilizador.

A partir do momento em que o LUIS começa a ter pedidos, as intents e as entities que não tenham uma grande percentagem de certeza podem ser aprimoradas de forma a que a performance do LUIS seja cada vez maior. Na figura abaixo é apresentado o processo cíclico de aprendizagem do LUIS. [7,8]

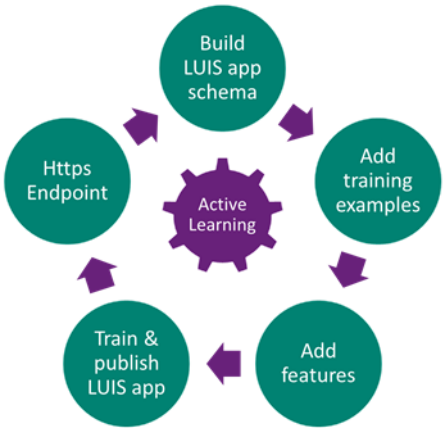

Figura 3 - Machine Learning in LUIS [8]

Após o desenho do LUIS, treino e publicação, ele estará preparado para começar a receber e enviar mensagens num formato JSON através de pedidos HTTP. Nas duas figuras abaixo é demonstrado como este serviço funciona.

Figura 4 - Identify Intent and Entities [7]

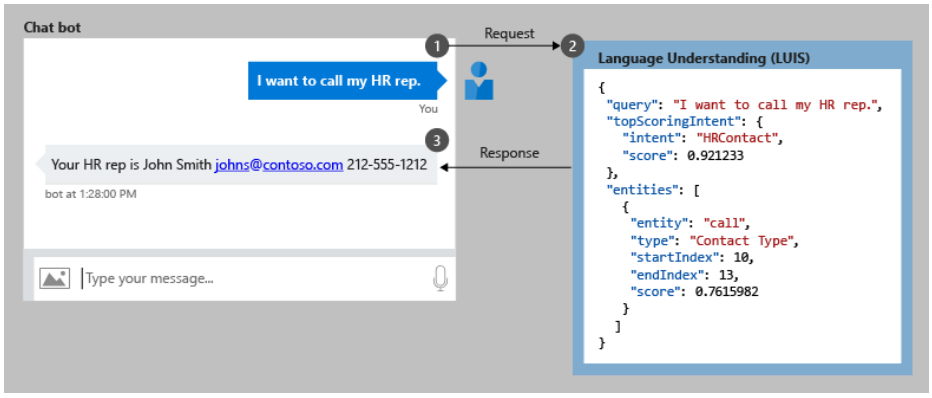

Figura 5 - Identificação e Processamento das intents e entities

2.3.4 Qna Maker (Base de Conhecimento)

Este é mais um dos serviços que a *Microsoft Azure* disponibiliza. Esta ferramenta permite ao fazer upload de ficheiros de texto, *urls*, entre outros, que vão ser usados como base de conhecimento para um sistema de Faqs.

Este sistema torna mais eficiente o Assistente Virtual, porque ao invés de ser necessário processar uma mensagem de resposta para o Cliente, a resposta é gerada automaticamente, a partir das repostas já pré-formuladas na base de conhecimento até então enriquecida.

2.3.5 Verificação de Ortografia e Análise de Sentimentos

A *Microsoft Azure*, possui diversos serviços, chamados serviços cognitivos, com soluções que oferecem diversos algoritmos para que as web apps e *bots*, ganhem capacidades de visão, voz, linguagem e conhecimento.

Para a correção de ortografia, são usadas estatísticas do motor de busca Bing, de forma a corrigir algum erro ortográfico que o utilizador possa ter cometido na escrita da mensagem. Este serviço possui ainda capacidades de análise a nível de nomes escritos incorretamente, separação errada de palavras e palavras homónimas. Assim, é possível que o Assistente Virtual ao receber uma mensagem com que contenha erros, consiga na mesma entendê-la.

A análise de sentimentos é um serviço que através das mensagens enviadas pelo Cliente, consegue determinar uma percentagem de felicidade ou de infelicidade do mesmo. Com esta avaliação da mensagem é possível, então, realizar ações que são consequências do estado de humor do Cliente.

2.4 AOP (Programação Orientada a Aspetos)

A Programação Orientada a Aspetos é uma técnica proposta para melhorar a separação de contextos, não sendo necessário a alteração da estrutura base de um sistema de software para que seja implementada uma solução. Este tipo de abordagem pode ser interpretado como um complemento às linguagens procedimentais e orientadas a objetos, cujo objetivo é intersectar classes ou funções/métodos em tempo de execução para realizar ações secundárias no sistema.

Existem diversas *frameworks* e extensões que auxiliam o nosso *software* a realizar ações com esta abordagens, mas do ponto de vista do nosso sistema, desenvolvido em *typescript*, foi decidido utilizar uma biblioteca denominada de *aspect.js*, que possui uma documentação bastante simples de utilizar e que quando comparada com outras

existentes, esta foi a única que conseguiu responder às necessidades que este assistente virtual endereça.

Na programação orientada a aspetos, um aspeto é uma classe que contem os métodos que vão ser executados mediante a ocorrência de alguma ação predefinida. A estas ocorrências são chamadas de *Join Points*, que de acordo com a biblioteca utilizada, que corresponde a uma anotação, é onde se define a classe e o método que se pretende intersectar em tempo de execução. Esta anotação é colocada nestes métodos definidos na classe aspeto.

As três principais anotações são: *@beforeMethod*, *@afterMethod* e *@aroundMethod*. Estas anotações, são utilizadas respetivamente, antes, depois e durante a execução de um programa para modificar ou adicionar possíveis ações em diferentes momentos da execução. Se quisermos alterar o retorno de um método, é usada a anotação *@aroundMethod* e dentro do método definido podemos adulterar, por exemplo, a mensagem de retorno.

No contexto de programação orientada a aspetos, também existem os *PointCuts*, que são os predicados correspondentes aos *JoinPoints*. Mas na biblioteca usada este termo não é tido em conta. [26,27,28]

2.5 V-Model (Modelo Verificação e Validação)

Na maioria dos projetos de desenvolvimento de software, sejam curtos ou longos, existe sempre uma metodologia com o objetivo de otimizar e melhorar a forma como se faz o seu desenho, o mesmo é desenvolvido e testado para que o projeto seja realizado com a máxima qualidade.

De acordo com cada projeto de desenvolvimento, existem diferenças e preferências na escolha de qual o modelo a selecionar. Mediante o tempo e o tipo de projeto a ser desenvolvido, existirá uma metodologia mais apropriada.

Existem vários modelos entre os quais podemos enumerar: *Waterfall*, Iterativo, *V-Model*, *Agile*, etc. Neste presente projeto foi escolhido o modelo *V-Model*, uma extensão do modelo *Waterfall*, em que a execução das tarefas é feita sequencialmente em forma de um ‘V’.

Nesta metodologia cada fase de desenvolvimento é correspondida por uma fase de testes, o que faz com que esta metodologia seja bastante disciplinada, na medida em que cada fase só começa quando a fase anterior esteja completa.

Como podemos ver na figura em baixo em forma de 'V', o lado esquerdo pertence as fases de Verificação e do lado direito às fases da Validação, sendo estas fases planeadas em paralelo. As de Verificação são: *Requirements Design, System Design, Architecture Design* e *Component Design Models*. De seguida existe a fase da Implementação da

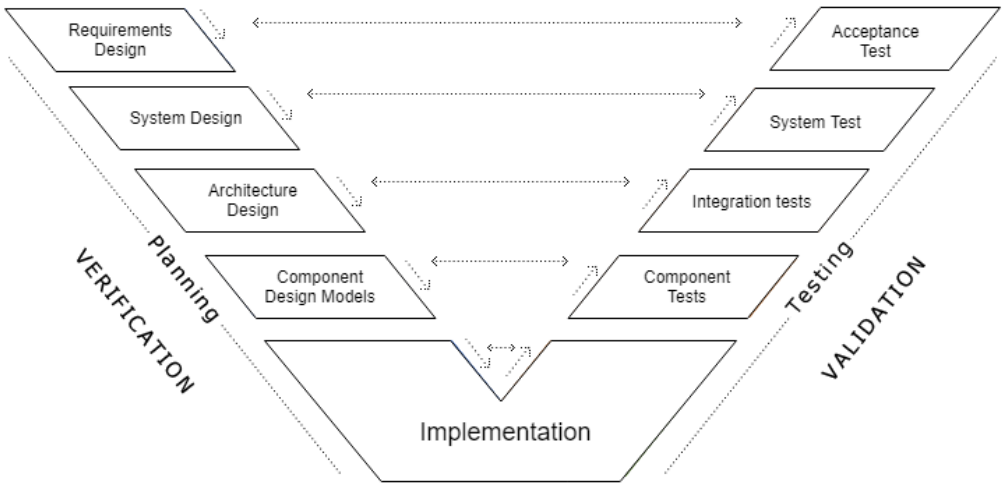

Figura 6 - Modelo em V

solução, e no final do ciclo, temos a fase de Validação, composta por: *Acceptance Test*, *System Test*, *Integration Test* e *Component Test*. [20,21]

Fases de Verificação (Desenvolvimento):

1. *Requirement Design*: Esta primeira fase é onde se define os requisitos através da conversação com o Cliente, ou seja, onde se percebe o que o Cliente quer de facto. É nesta fase também que se desenha o plano da fase *Acceptance Test.*
2. *System Design*: Depois de ter os requisitos bem definidos do produto, é nesta fase que se vai desenhar o sistema. Desenhar o sistema, significa perceber e detalhar que hardware e que tipo de comunicações vamos configurar no sistema. Nesta fase também se planeiam os testes a realizar ao mesmo.
3. *Architecture Design* – As especificações da arquitetura são definidas e desenhadas (Desenho de Alto Nível) nesta fase, que por norma são propostas mais que uma abordagem e só depois se decide pela melhor. Neste momento também se definem os testes de Integração.
4. *Component Design Models* – Nesta Fase são realizados os desenhos detalhados para todos os módulos internos do sistema (Desenho de Baixo Nível). É importante que este desenho seja feito e compatível com os outros

módulos da arquitetura do sistema ou com outros módulos externos. É também nesta fase que se realiza o plano de testes correspondente.

Fase de Implementação:

Após definidas as tecnologias a utilizar, é feito o desenvolvimento do projeto, onde também podem existir pequenas fases para realização de testes unitários a certas componentes do código. Depois do projeto estar implementado é iniciada a fase de Validação.

Fase de Validação:

Todos os testes são realizados com base nos planeamentos realizados na fase de verificação.

1. Component Test – Estes testes são planeados para serem executados com a finalidade de detetar erros e eliminá-los nos componentes de baixo nível.
2. *Integration Test* - Testes direcionados para testar a comunicação dos módulos internos dentro do sistema.
3. *System Test* – os testes realizados nesta fase têm que testar as funcionalidades do sistema e a comunicação com sistemas externos.
4. *Acceptance Test* – Esta é a última fase de realização de testes ao sistema, onde se vai validar se o nosso sistema corresponde aos requisitos inicialmente definidos.

Capítulo 3 Planeamento

Este capítulo encontra-se omisso por confidencialidade.

Capítulo 4 Desenho

Este capítulo encontra-se omisso por confidencialidade.

Capítulo 5 Desenvolvimento

Este capítulo encontra-se omisso por confidencialidade.

Capítulo 6 Testes

Este capítulo encontra-se omisso por confidencialidade.

Capítulo 7 Conclusão

7.1 Dificuldades Encontradas

Este capítulo encontra-se omisso por confidencialidade.

7.2 Conclusão do Projeto

Este Projeto de Engenharia Informática foi desenvolvido na empresa Accenture Portugal, com o objetivo de implementar uma plataforma na área de faturação, para auxiliar os Clientes de empresas de telecomunicações, como mais uma alternativa de apoio ao Cliente que permita a estes Clientes um acesso rápido a respostas às suas questões, sem a necessidade de esperar em filas ou ter que ligar para um *call-center*.

A solução criada é uma aplicação web onde é integrado um Assistente Virtual, que vem humanizar esta aplicação, através das suas capacidades de conversação com o Cliente, destacando-se de outras assistentes já existentes, por ter sido desenvolvido com o propósito, de resolver um problema, que desde sempre esteve endereçado nas faturas do Cliente. Adicionando assim, um valor acrescentado a este Assistente Virtual.

Para tornar a implementação deste Assistente Virtual possível, recorreu-se a tecnologias disponibilizadas pela Microsoft, mais especificamente ao Bot Framework, LUIS e a Table Storage. Através destas tecnologias, fomos obrigados a seguir uma arquitetura, que quando adaptada à sua estrutura para o contexto do Cliente, resultou na solução final do Assistente Virtual que foi apresentado no presente documento.

Com os resultados e análises dos testes ao NLP, testes ao sistema de análise da fatura, testes de Integração e os testes de aceitação, que foram maioritariamente positivos, posso concluir que este Assistente Virtual desenvolvido para ser uma prova de conceito, teve sucesso e que apesar das limitações idenficadas, este pode ser muito eficaz e determinante na forma como as empresas podem responder às solicitações dos seus clientes, simplificando através de automatismos a comunicação entre o cliente e as mesmas, melhorando a experência do cliente e qualidade da informação transmitida.

7.3 Trabalho Futuro

Este capítulo encontra-se omisso por confidencialidade.

Bibliografia

[1] Friedes, Albert. "Interactive queuing system for call centers." U.S. Patent No. 5,444,774. 22 Aug. 1995.

[2] Ayesha Shaikh, Geetanjali Phalke, Pranita Patil, Sangita Bhosale, Jyoti Raghatwan, “Chatbot : Artificially Intelligent Conversational Agent”

[3] Shawar, Bayan Abu, and Eric Atwell. "Chatbots: are they really useful?." Ldv Forum. Vol. 22. No. 1. 2007.

[4] Karray, Fakhreddine, et al. "Human-computer interaction: Overview on state of the art." (2008).

[5] Microsoft Azure Documentation [Online] Available: https://docs.microsoft.com/pt-pt/azure/#pivot=get-started [Acedido em 7 de Maio de 2019]

[6] Funcionamento Bot Framework Service [Online] Available: https://docs.microsoft.com/en-us/azure/bot-service/bot-service-overview-introduction?view=azure-bot-service-4.0 [Acedido em 7 de Maio de 2019]

[7] Funcionamento LUIS [Online] Available: https://azure.microsoft.com/en-us/blog/luis-ai-automated-machine-learning-for-custom-language-understanding/ [Acedido em 7 de Maio de 2019]

[8] Funcionamento do LUIS [Online] Available: https://docs.microsoft.com/en-us/azure/cognitive-services/luis/what-is-luis [Acedido em 7 de Maio de 2019]

[9] Funcionamento do QnA Maker [Online] Available: https://docs.microsoft.com/en-us/azure/cognitive-services/QnAMaker/overview/overview [Acedido em 8 de Maio de 2019]

[10] Node.js [Online] Available: https://nodejs.org/en/about/ [Acedido em 5 de Maio de 2019]

[11] Node.js javascript [Online] Available: https://itnext.io/how-javascript-works-in-browser-and-node-ab7d0d09ac2f [Acedido em 5 de Maio de 2019]

[12] Node.js and java script Non-blocking or blocking [Online] Available: https://www.codementor.io/theresamostert/understanding-non-blocking-i-o-in-javascript-cvmg1hp6l [Acedido em 5 de Maio de 2019]

[13] NPM [Online] Available: https://docs.npmjs.com/getting-started/what-is-npm [Acedido em 5 de Maio de 2019]

[14] C# Linguagem de Programação [Online] Available: https://docs.microsoft.com/en-us/dotnet/csharp/tour-of-csharp/index [Acedido em 6 de Maio de 2019]

[15] C# Microsoft guide [Online] Available: https://docs.microsoft.com/en-us/dotnet/csharp/ [Acedido em 6 de Maio de 2019]

[16] Funcionamento Amazon Lex [Online] Available:

https://docs.aws.amazon.com/lex/latest/dg/what-is.html [Acedido em 10 de Janeiro de 2019]

[17] Limites Amazon Lex [Online] Available:

https://docs.aws.amazon.com/lex/latest/dg/gl-limits.html [Acedido em 10 de Janeiro de 2019]

[18] Funcionamento Dialogflow [Online] Available: https://dialogflow.com/docs/intro [Acedido em 10 de Janeiro de 2019]

[19] Funcionamento IBM Watson Assistant [Online] Available:

https://console.bluemix.net/docs/services/conversation/index.html#about [Acedido em 10 de Janeiro de 2019]

[20] V – Model [Online] Available http://www.professionalqa.com/v-model [Acedido em 12 de Fevereiro de 2019]

[21]V-Model [Online] Available:

http://moodle.autolab.unipannon.hu/Mecha_tananyag/szoftverfejlesztesi _folyamatok_angol/ch03.html#d0e550 [Acedido em 12 de Fevereiro de 2019]

[22] Javascript [Online] Available: https://javascript.info/getting-started [Acedido em 6 de Maio de 2019]

[23] Typescript [Online] Available: https://www.typescriptlang.org/index.html [Acedido em 14 de Janeiro de 2019]

[24] Microsoft Cognitive Services [Online] Available: https://azure.microsoft.com/pt-pt/services/cognitive-services/ [Acedido em 15 de Abril de 2019]

[25] Azure Table Storage [Online] Available: https://docs.microsoft.com/en-us/azure/cosmos-db/table-storage-overview [Acedido em 10 de Maio de 2019]

[26] AOP Package [Online] Available: https://www.npmjs.com/package/aspect.js [Acedido em 15 de Maio de 2019]

[27] Gregor Kiczales, James Hugunin, Erik Hilsdale, Mik Kersten, Jeff Palm, Crista Lopes, Bill Griswold, and Wes Isberg, “Aspect Oriented Programming”, Palo Alto Research Center, July 2003

[28] Kiczales, Gregor, et al. "Aspect-oriented programming." European conference on object-oriented programming. Springer, Berlin, Heidelberg, 1997.